

14-5 Edit distance

In order to transform a source string of text $x[1:m]$ to a target string $y[1:n]$, you can perform various transformation operations. The goal is, given x and y , to produce a series of transformations that changes x to y . An array z —assumed to be large enough to hold all the characters it needs—holds the intermediate results. Initially, z is empty, and at termination, you should have $z[j] = y[j]$ for $j = 1, 2, \dots, n$. The procedure for solving this problem maintains current indices i into x and j into z , and the operations are allowed to alter z and these indices. Initially, $i = j = 1$. Every character in x must be examined during the transformation, which means that at the end of the sequence of transformation operations, $i = m + 1$.

You may choose from among six transformation operations, each of which has a constant cost that depends on the operation:

Copy a character from x to z by setting $z[j] = x[i]$ and then incrementing both i and j . This operation examines $x[i]$ and has cost Q_C .

Replace a character from x by another character c , by setting $z[j] = c$, and then incrementing both i and j . This operation examines $x[i]$ and has cost Q_R .

Delete a character from x by incrementing i but leaving j alone. This operation examines $x[i]$ and has cost Q_D .

Insert the character c into z by setting $z[j] = c$ and then incrementing j , but leaving i alone. This operation examines no characters of x and has cost Q_I .

Twiddle (i.e., exchange) the next two characters by copying them from x to z but in the opposite order: setting $z[j] = x[i + 1]$ and $z[j + 1] = x[i]$, and then setting $i = i + 2$ and $j = j + 2$. This operation examines $x[i]$ and $x[i + 1]$ and has cost Q_T .

Kill the remainder of x by setting $i = m + 1$. This operation examines all characters in x that have not yet been examined. This operation, if performed, must be the final operation. It has cost Q_K .

Figure 14.12 gives one way to transform the source string `algorithm` to the target string `altruistic`. Several other sequences of transformation operations can transform `algorithm` to `altruistic`.

Assume that $Q_C < Q_D + Q_I$ and $Q_R < Q_D + Q_I$, since otherwise, the copy and replace operations would not be used. The cost of a given sequence of transformation operations is the sum of the costs of the individual operations in the sequence. For the sequence above, the cost of transforming `algorithm` to `altruistic` is $3Q_C + Q_R + Q_D + 4Q_I + Q_T + Q_K$.

a. Given two sequences $x[1:m]$ and $y[1:n]$ and the costs of the transformation operations, the *edit distance* from x to y is the cost of the least expensive op-

Operation	<i>x</i>	<i>z</i>
<i>initial strings</i>	<u>a</u> lgorithm	_
copy	a <u>l</u> gorithm	a_
copy	al <u>g</u> orithm	al_
replace by t	alg <u>o</u> rithm	alt_
delete	algor <u>i</u> thm	alt_
copy	algori <u>t</u> hm	altr_
insert u	algori <u>u</u> thm	altru_
insert i	algori <u>i</u> thm	altrui_
insert s	algori <u>s</u> thm	altruiss_
twiddle	algori <u>u</u> s <u>t</u> hm	altruist <u>i</u> _
insert c	algori <u>u</u> s <u>t</u> h <u>m</u>	altruistic_
kill	algorithm_	altruistic_

Figure 14.12 A sequence of operations that transforms the source `algorithm` to the target string `altruistic`. The underlined characters are $x[i]$ and $z[j]$ after the operation.

eration sequence that transforms x to y . Describe a dynamic-programming algorithm that finds the edit distance from $x[1:m]$ to $y[1:n]$ and prints an optimal operation sequence. Analyze the running time and space requirements of your algorithm.

The edit-distance problem generalizes the problem of aligning two DNA sequences (see, for example, Setubal and Meidanis [405, Section 3.2]). There are several methods for measuring the similarity of two DNA sequences by aligning them. One such method to align two sequences x and y consists of inserting spaces at arbitrary locations in the two sequences (including at either end) so that the resulting sequences x' and y' have the same length but do not have a space in the same position (i.e., for no position j are both $x'[j]$ and $y'[j]$ a space). Then we assign a “score” to each position. Position j receives a score as follows:

- +1 if $x'[j] = y'[j]$ and neither is a space,
- −1 if $x'[j] \neq y'[j]$ and neither is a space,
- −2 if either $x'[j]$ or $y'[j]$ is a space.

The score for the alignment is the sum of the scores of the individual positions. For example, given the sequences $x = \text{GATCGGCAT}$ and $y = \text{CAATGTGAATC}$, one alignment is

```
G ATCG GCAT
CAAT GTGAATC
- * + + * + * - + + *
```

A + under a position indicates a score of +1 for that position, a – indicates a score of –1, and a * indicates a score of –2, so that this alignment has a total score of $6 \cdot 1 - 2 \cdot 1 - 4 \cdot 2 = -4$.

- b.* Explain how to cast the problem of finding an optimal alignment as an edit-distance problem using a subset of the transformation operations copy, replace, delete, insert, twiddle, and kill.

14-6 Planning a company party

Professor Blutarsky is consulting for the president of a corporation that is planning a company party. The company has a hierarchical structure, that is, the supervisor relation forms a tree rooted at the president. The human resources department has ranked each employee with a conviviality rating, which is a real number. In order to make the party fun for all attendees, the president does not want both an employee and his or her immediate supervisor to attend.

Professor Blutarsky is given the tree that describes the structure of the corporation, using the left-child, right-sibling representation described in Section 10.3. Each node of the tree holds, in addition to the pointers, the name of an employee and that employee's conviviality ranking. Describe an algorithm to make up a guest list that maximizes the sum of the conviviality ratings of the guests. Analyze the running time of your algorithm.

14-7 Viterbi algorithm

Dynamic programming on a directed graph can play a part in speech recognition. A directed graph $G = (V, E)$ with labeled edges forms a formal model of a person speaking a restricted language. Each edge $(u, v) \in E$ is labeled with a sound $\sigma(u, v)$ from a finite set Σ of sounds. Each directed path in the graph starting from a distinguished vertex $v_0 \in V$ corresponds to a possible sequence of sounds produced by the model, with the label of a path being the concatenation of the labels of the edges on that path.

- a.* Describe an efficient algorithm that, given an edge-labeled directed graph G with distinguished vertex v_0 and a sequence $s = \langle \sigma_1, \sigma_2, \dots, \sigma_k \rangle$ of sounds from Σ , returns a path in G that begins at v_0 and has s as its label, if any such path exists. Otherwise, the algorithm should return NO-SUCH-PATH. Analyze the running time of your algorithm. (*Hint:* You may find concepts from Chapter 20 useful.)

Now suppose that every edge $(u, v) \in E$ has an associated nonnegative probability $p(u, v)$ of being traversed, so that the corresponding sound is produced. The sum of the probabilities of the edges leaving any vertex equals 1. The probability of a path is defined to be the product of the probabilities of its edges. Think of